**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 27.10.2023 |
| Team ID | Team- 590871 |
| Project Name | Owl-M : A Material Design Study App |
| Maximum Marks | 4 |

**Table-1 : Components & Technologies:**

|  |  |
| --- | --- |
| Component | Technology Used |
| User Interface | Compose or Material Components for Android |
| Application Logic-1 | Manages user accounts (create, login, update) |
| Application Logic-2 | Tracks user progress and provides feedback |
| Application Logic-3 | Generates study sessions based on user progress and materials |
| Database | Local database (SQLite) for storing user data, materials, and study sessions |
| Cloud Database | Remote API (Firebase) for syncing user data and materials across devices |
| File Storage | Local file storage for storing user-generated files (e.g., notes, study plans) |
| External API-1 | Connects to a third-party API for retrieving additional materials |
| External API-2 | Connects to a third-party API for personalized study recommendations |
| Machine Learning Model | None |
| Infrastructure | Android Studio, IntelliJ IDEA, or Visual Studio |
| Target Platform | Android 12.0 or higher |
| App Size | Less than 100MB |
| Memory Usage | Less than 1GB |
| CPU Usage | Less than 10% |
| Network Usage | Less than 1MB/session |

**Table-2: Application Characteristics:**

|  |  |
| --- | --- |
| Component | Characteristics |
| Open-Source Frameworks | Compose or Material Components for Android (UI framework), Retrofit or Volley (HTTP client library), Room or Realm (local database library), Firebase Realtime Database or Cloud Firestore (cloud database) |
| Security Implementations | User authentication with secure password hashing, data encryption at rest and in transit, regular vulnerability scanning and patching |
| Scalable Architecture | Use cloud-based services for data storage and syncing (Firebase) to handle increasing user base and data volume, implement modular design with well-defined interfaces for easy scaling |
| Availability | Monitor app uptime and performance metrics, implement redundancy in infrastructure (e.g., load balancers, failover servers) to ensure high availability |
| Performance | Optimize app performance by using efficient algorithms and data structures, avoid unnecessary data processing and network requests, utilize caching mechanisms |